ИУ5-34Б
Юрченко Кирилл

Отчёт РК2 по дисциплине “Парадигмы и конструкции языков программирования”

Тест для запроса 1
Тест проверяет корректность выполнения запроса, который возвращает список всех книжных магазинов, у которых название начинается на "А", и список книг в этих магазинах. Ожидаемый результат сравнивается с фактическим результатом выполнения функции query_1.

Тест для запроса 2
Тест проверяет корректность выполнения запроса, который возвращает список магазинов, отсортированный по цене самых дорогих книг и стоимость этих книг. Ожидаемый результат сравнивается с фактическим результатом выполнения функции query_2.

Тест для запроса 3
Тест проверяет корректность выполнения запроса, который возвращает список всех магазинов, а также список всех книг, продающихся в этих магазинах. Ожидаемый результат сравнивается с фактическим результатом выполнения функции query_3.

Текст main.py:

```
from operator import itemgetter

    class Book:
        def __init__(self, id, title, price):
            self.id = id
            self.title = title
            self.price = price

    class Bookstore:
        def __init__(self, id, name):
            self.id = id
            self.name = name

    class BookStoreLink:
        def __init__(self, store_id, book_id):
            self.store_id = store_id
            self.book_id = book_id

    def create_one_to_many(bookstores, book_store_links, books):
        return [(b.title, b.price, s.name)
                for s in bookstores
                for link in book_store_links
                for b in books
                if link.store_id == s.id and link.book_id == b.id]

    def query_1(bookstores, one_to_many):
        return {s.name: [book[0] for book in one_to_many if book[2] ==
    s.name]
                for s in bookstores if s.name.startswith("A")}

    def query_2(bookstores, one_to_many):
        res_2_unsorted = []
        for s in bookstores:
            s_books = list(filter(lambda i: i[2] == s.name, one_to_many))
            if s_books:
                s_max_price = max([price for _, price, _ in s_books])
                res_2_unsorted.append((s.name, s_max_price))
        return sorted(res_2_unsorted, key=itemgetter(1), reverse=True)

    def query_3(bookstores, book_store_links, books):
        return {store.name: [book.title for link in book_store_links for
    book in books
                             if link.store_id == store.id and link.book_id
    == book.id]
                for store in bookstores}

    bookstores = [
        Bookstore(1, 'Альфа книги'),
        Bookstore(2, 'Бета книги'),
        Bookstore(3, 'Академия знаний')
    ]
    books = [
        Book(1, 'Python для начинающих', 500),
```

```
        Book(2, 'Алгоритмы и структуры данных', 700),
        Book(3, 'Машинное обучение', 1200),
        Book(4, 'Анализ данных', 800),
        Book(5, 'Основы программирования', 550)
    ]

    book_store_links = [
        BookStoreLink(1, 1),
        BookStoreLink(1, 2),
        BookStoreLink(2, 3),
        BookStoreLink(3, 4),
        BookStoreLink(3, 5),
        BookStoreLink(1, 4),
        BookStoreLink(1, 3),
        BookStoreLink(2, 2),
        BookStoreLink(3, 1),
    ]

    def main():
        one_to_many = create_one_to_many(bookstores, book_store_links,
    books)

        print("Запрос 1")
        res_1 = query_1(bookstores, one_to_many)
        for store, titles in res_1.items():
            print(f"{store}: {', '.join(titles)}.")

        print("\nЗапрос 2")
        res_2 = query_2(bookstores, one_to_many)
        for store, max_price in res_2:
            print(f"{store}: {max_price} руб.")

        print("\nЗапрос 3")
        res_3 = query_3(bookstores, book_store_links, books)
        for store, titles in res_3.items():
            print(f"{store}: {', '.join(titles)}.")

    if __name__ == '__main__':
        main()
```

Текст test_main.py:

```
import unittest
    from main import create_one_to_many, query_1, query_2, query_3,
    bookstores, books, book_store_links

    class TestBookstoreQueries(unittest.TestCase):

        def setUp(self):
            self.bookstores = bookstores
            self.books = books
            self.book_store_links = book_store_links
            self.one_to_many = create_one_to_many(self.bookstores,
    self.book_store_links, self.books)

        def test_query_1(self):
            expected = {
                'Альфа книги': ['Python для начинающих', 'Алгоритмы и
    структуры данных', 'Анализ данных', 'Машинное обучение'],
                'Академия знаний': ['Анализ данных', 'Основы
    программирования', 'Python для начинающих']
            }
            result = query_1(self.bookstores, self.one_to_many)
            self.assertEqual(result, expected)

        def test_query_2(self):
            expected = [
                ('Альфа книги', 1200),
                ('Бета книги', 1200),
                ('Академия знаний', 800)
            ]
            result = query_2(self.bookstores, self.one_to_many)
            self.assertEqual(result, expected)

        def test_query_3(self):
            expected = {
                'Альфа книги': ['Python для начинающих', 'Алгоритмы и
    структуры данных', 'Анализ данных', 'Машинное обучение'],
                'Бета книги': ['Машинное обучение', 'Алгоритмы и структуры
    данных'],
                'Академия знаний': ['Анализ данных', 'Основы
    программирования', 'Python для начинающих']
            }
            result = query_3(self.bookstores, self.book_store_links,
    self.books)
            self.assertEqual(result, expected)

    if __name__ == '__main__':
        unittest.main()
```

Результат выполнения программы:

```
Testing started at 02:24 ...


Ran 3 tests in 0.001s
Launching unittests with arguments python -m unittest D:\Study\projects\RK2-PCPL\test_main.py in D:\Study\projects\RK2-PCPL


OK

Process finished with exit code 0
```